Discussion of Coupled Modeling

Dan Schaffer

NOAA Forecast Systems Laboratory

October 2003

Outline

- Coupling Methodologies
- Tools
- Coupled WRF/ROMS implementation and results
- Conclusions

Coupling Methodologies

- Introduction
  - Many design decisions required
  - Often, each decision is orthogonal to the others
  - Choices made governed by
    - Scientific needs
    - Available funding/time

Coupling Methodologies (contd)

- Choices include:
  - Code structure:
    Subroutine vs. Component
  - Component interaction:
    Scheduled vs. Producer/Consumer
  - Component execution:
    Sequential vs. Concurrent
  - Coupler design:
    Hub and Spokes vs. Tinker Toy
  - Coupling execution:
    Inside Model vs. Outside Model

Code Structure

- Subroutinize method (MM5 chemistry)
  - Comm via subroutine calls and argument lists
  - Modules must be language-compatible
  - Requires that one model be "in control"
  - More efficient
- Example

```
subroutine run_atmos
  call ocean_model(Wind)
  call land_model(Wind)
```

Code Structure (contd)

- Component method (CCSM)
  - Communication: through I/O or via messages
  - Component interaction: Scheduled or Producer/Consumer
  - Component execution: Sequential or Concurrent
  - More flexible

- Example

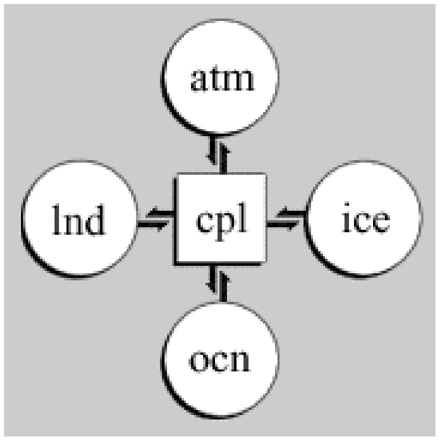
```
subroutine run_atmos
  call send_msg(Wind, to_ocean_model)
  call send_msg(Wind, to_land_model)
```

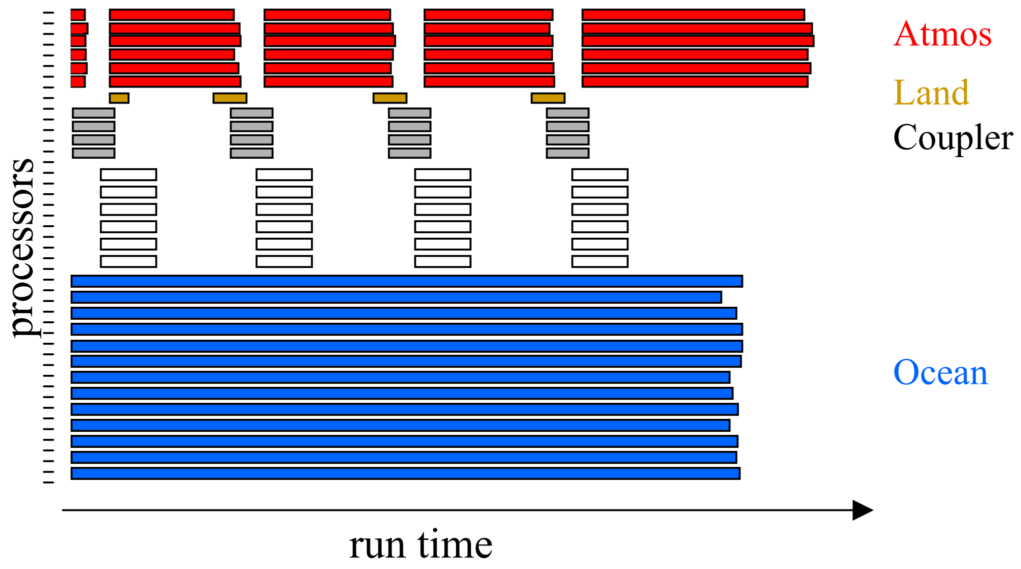
Component interaction

- Scheduled (CCSM)
  - Regular, pre-determined interaction schedule
  - Centrally Controlled
- Producer/Consumer (HFSoLE)
  - No pre-determined schedule
  - No central control
  - Components initiate data production and consumption on their own schedule

Component Execution: Concurrent

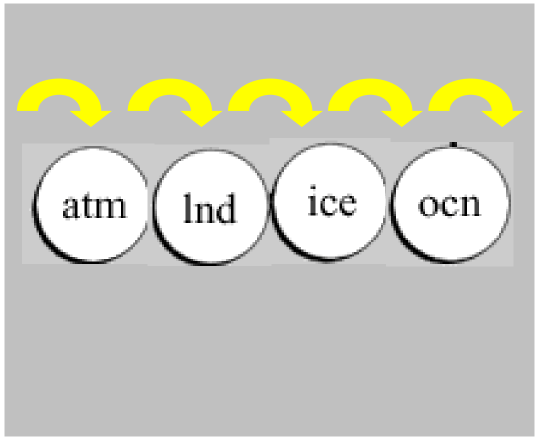
- Concurrent coupling (CCSM)
  - Components integrate concurrently on separate sets of processes
  - Periodically communicate forcing data to other components on some schedule
  - Parallelism is both within and between the components; subject to load imbalance
  - Two-way coupling requires solutions from components to lag

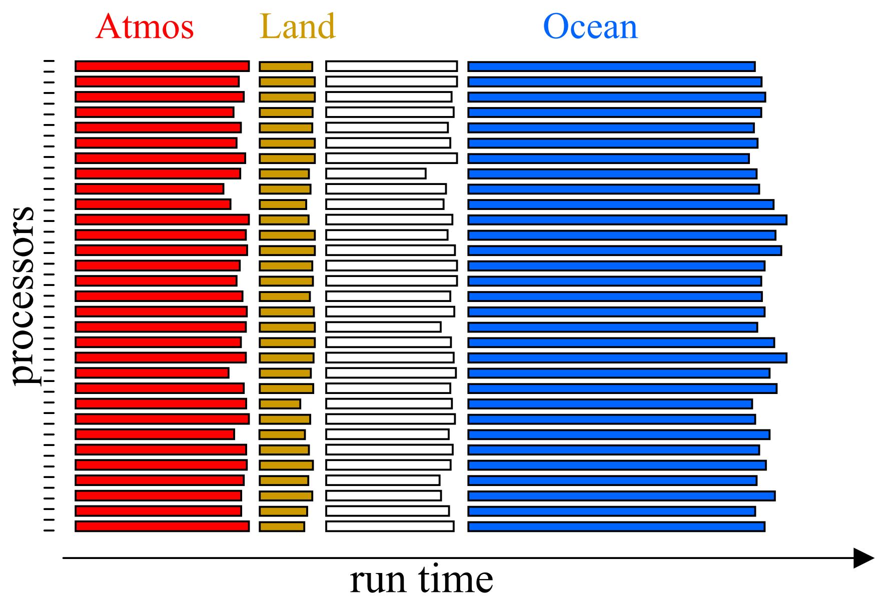




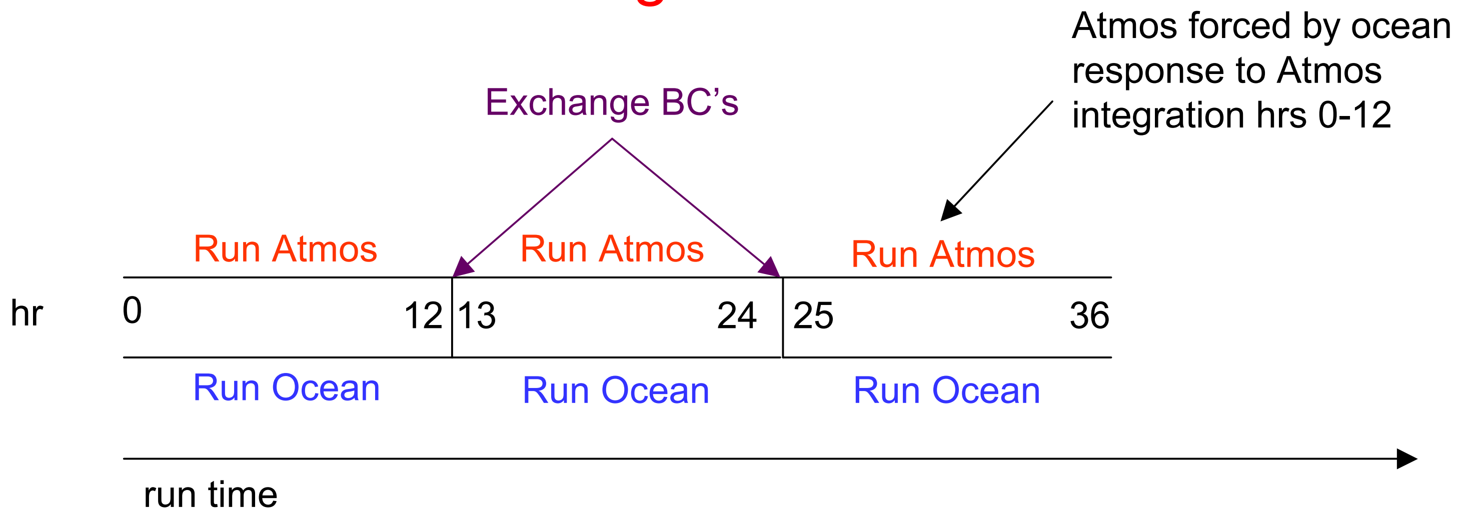
Component Execution: Sequential

- Sequential coupling (PCM)
  - Components are integrated in sequence by all processors
  - All processes can be kept active without forcing components to lag
  - Performance and scaling is limited by least scalable component

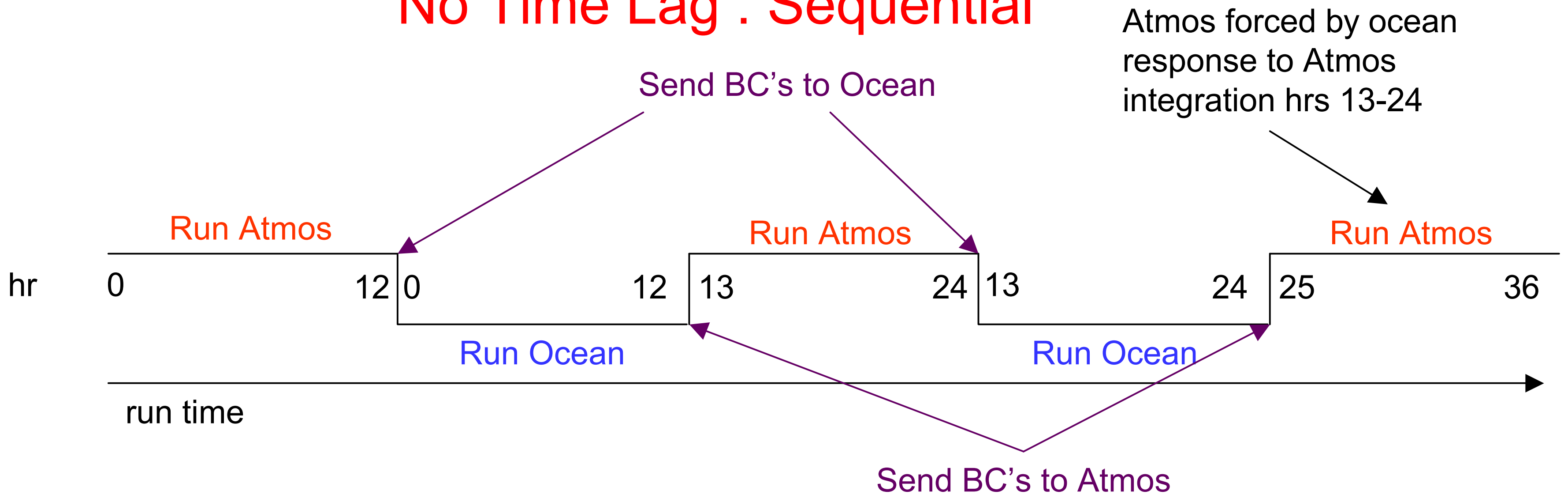




Time Lag : Concurrent



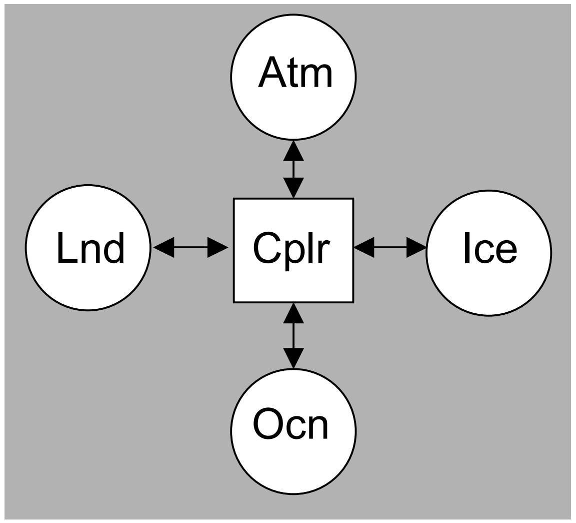
No Time Lag : Sequential



Coupler Design : Hub and Spoke

- Advantage
  - Each component needs to know about only one other piece : the coupler
- Disadvantages
  - Boundary conditions serialized through coupler bottleneck
  - Extra communication/interpolation is required (i.e. from atm to cpl to ocn)

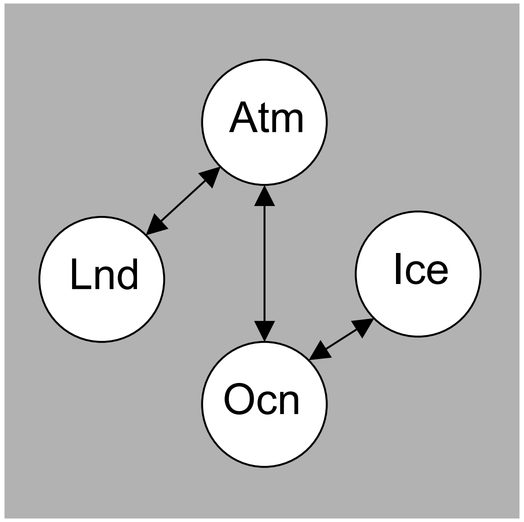
CCSM



Coupler Design : Tinker Toy

- Advantages
  - Only one communication is required between any component pair (i.e. atm to ocn)
  - Communication between component pairs can be done in parallel
- Disadvantage
  - Each component has to know grid and decomposition of every component with which it communicates

GEMS



Coupling Execution : Within the model (WRF/ROMS)

```
program atmos_ocean
do I = 1, NSTEPS
  if (I_AM_ATM_PE()) then
    call RUN_ATMOS ! calls couple_to_ocn
  else
    call RUN_OCEAN ! calls couple_to_atm
  end if
end do
```

- Advantage : Small impact on uncoupled code; replace “read_sst”, “write_wind_stress” with “couple_to_ocn”

Coupling Execution : Outside the model (GEMS)

```
program atmos_ocean
do I = 1, NSTEPS
  if (I_AM_ATM_PE()) then
    call RUN_ATMOS(ATMOS_STATE)
    call COUPLE_ATM_TO_OCN(ATMOS_STATE)
  else
    call RUN_OCEAN(OCEAN_STATE)
    call COUPLE_OCN_TO_ATM(OCEAN_STATE)
  end if
end do
```

- Advantage : Clean design; models don’t have to know when or how to couple

Tools

- Earth System Modeling Framework (ESMF) – Widespread Community Effort
- Multi-Component Handshaking Library (MPH) - Lawrence Berkeley Laboratory
- Model Coupling Toolkit (MCT)

  - Argonne National Laboratory
- WRF I/O API MCT Coupling Implementation

  - NCAR MMM Division

ESMF

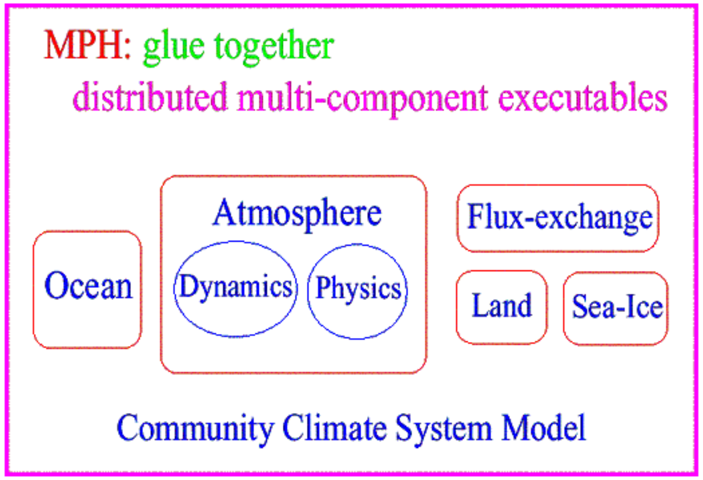
- NASA funded ($10 million/2002-2005) community effort
- Provides tools for ocean/atmos modeling at 2 levels
  - Super-structure: software to couple model components at a high level
  - Infra-structure: software to implement re-gridding, halo updates, nesting, etc.

ESMF (contd)

- Complex but potentially powerful software
- First full release in April 2004
- However, no promises made by developers regarding performance at that time
- Acceptance by the community is unclear
- FSL may eventually replace SMS libraries with ESMF
- www.esmf.ucar.edu

MPH: Multi-Component Handshaking Library

Integrate independent models into a simulation system



Main functionality

- ❑ Component name registration
- ❑ Mapping of PEs to components
- ❑ Query multi-component environment

Different model integration modes

- ❑ Single-Component Multi-Executable (SCME) CCSM
- ❑ Multi-Component Single-Executable (MCSE) PCM
- ❑ Multi-Component Multi-Executable (MCME) Most Flexible
- ❑ Multi-Instance Multi-Executable (MIME) Ensemble simulation

MPH: Multi-Component Handshaking Library

```
atm_ocn.F:
  mpi_comm = MPH_components (name1="atm", name2="ocn")

  if (PE_in_component("atm") then
    call run_atm
  end if
```

```
Registration File:
Multi-Comp-Start
  atm    0   11
  ocn   12   15
Multi-Comp-End
```

- Runs on IBM/SP, SGI/Origin, Compaq/SC, PC Clusters
- Users:
  - NCAR CCSM2.0.1
  - CSU Icosahedra grid coupled model
  - NOAA for coupling models over Grids
  - UK group, Germany group, SGI

Source codes, test examples, installation, available online:

http://www.nersc.gov/research/SCG/acpi/MPH

Model Coupling Toolkit

- Software tool set for creating a parallel coupled model
- Developed at Argonne National Laboratory
- Available in a Fortran 90 interface

Model Coupling Toolkit (contd)

- Provides the following services:
  - Domain decomposition descriptors
  - A flexible random-access field data storage data type
  - Communications schedulers for parallel inter-component data transfer and intra-component data redistribution
  - As part of these data transfers, inter-grid interpolation implemented as matrix-vector multiplications

Model Coupling Toolkit (contd)

- More services:
  - A time averaging and accumulation buffer data type
  - A general spatial grid representation capable of supporting unstructured grids

WRF I/O API MCT Coupling Implementation

- WRF I/O API abstracts input/output implementation details
- MCT coupling is an extension to the WRF I/O API specification
  - Coupling looks like I/O:
    - Input and output channels are "opened"
    - Receives correspond to reads; sends correspond to writes
  - Data are interpolated in parallel between component grids
  - "Training phase" used to construct MCT parallel interpolators
    - Communication patterns are cached away for efficiency

WRF MCT I/O API Key Subroutines

- `EXT_MCT_IOINIT`
- `EXT_MCT_OPEN_FOR_READ_BEGIN`
- `EXT_MCT_OPEN_FOR_WRITE_BEGIN`
- `EXT_MCT_OPEN_FOR_READ_COMMIT`
- `EXT_MCT_OPEN_FOR_WRITE_COMMIT`
- `EXT_MCT_READ_FIELD`
- `EXT_MCT_WRITE_FIELD`

API

SUBROUTINE ext_*mct*_ioinit(SysDepInfo, Status)

Synopsis:

Initialize the *Mct* coupler implementation of the WRF I/O API.

API

```
SUBROUTINE ext_mct_open_for_read_begin ( ComponentName , GlobalComm ,
                                         CompComm, SysDepInfo,
                                         DataHandle   , Status )
```

Synopsis:

Open a coupling stream in which the calling component will read data from *ComponentName*. Begin the “training” phase.

API

```
SUBROUTINE ext_mct_open_for_write_begin (ComponentName , GlobalComm ,
                                         CompComm, SysDepInfo,
                                         DataHandle   , Status )
```

Synopsis:

Open a coupling stream in which the calling component will write data to *ComponentName*. Begin the "training" phase.

API

```
SUBROUTINE ext_mct_open_for_read_commit( DataHandle , Status )

Synopsis:
       End “training” phase for the coupling stream referred to by
DataHandle.
```

API

```
SUBROUTINE ext_mct_open_for_write_commit( DataHandle , Status )
```

Synopsis:

End “training” phase for the coupling stream referred to by *DataHandle*.

API

```
SUBROUTINE ext_mct_read_field ( &
   DataHandle , DateStr , VarName ,       &
   Field , DimNames ,                     &
   DomainStart , DomainEnd ,              &
   MemoryStart , MemoryEnd ,              &
   PatchStart , PatchEnd )
```

Synopsis:

During “training”, construct and cache away data transfer communication patterns

Otherwise, receive the variable named *VarName* from the component pointed to by *DataHandle*. The data are stored into *Field*.

DomainStart and *DomainEnd* are the global starts and ends of the data.

MemoryStart and *MemoryEnd* are the PE local starts and ends including halo points.

PatchStart and *Patchend* are the PE local interior starts and ends.

API

```
SUBROUTINE ext_mct_write_field ( &
   DataHandle , DateStr , VarName ,       &
   Field , DimNames ,                     &
   DomainStart , DomainEnd ,              &
   MemoryStart , MemoryEnd ,              &
   PatchStart , PatchEnd )
```

Synopsis:

During "training", construct and cache away data transfer communication patterns

Otherwise, send the data for variable named *VarName* to the component pointed to by *DataHandle*.

DomainStart and *DomainEnd* are the global starts and ends of the data.

MemoryStart and *MemoryEnd* are the PE local starts and ends including halo points.

PatchStart and *Patchend* are the PE local interior starts and ends.

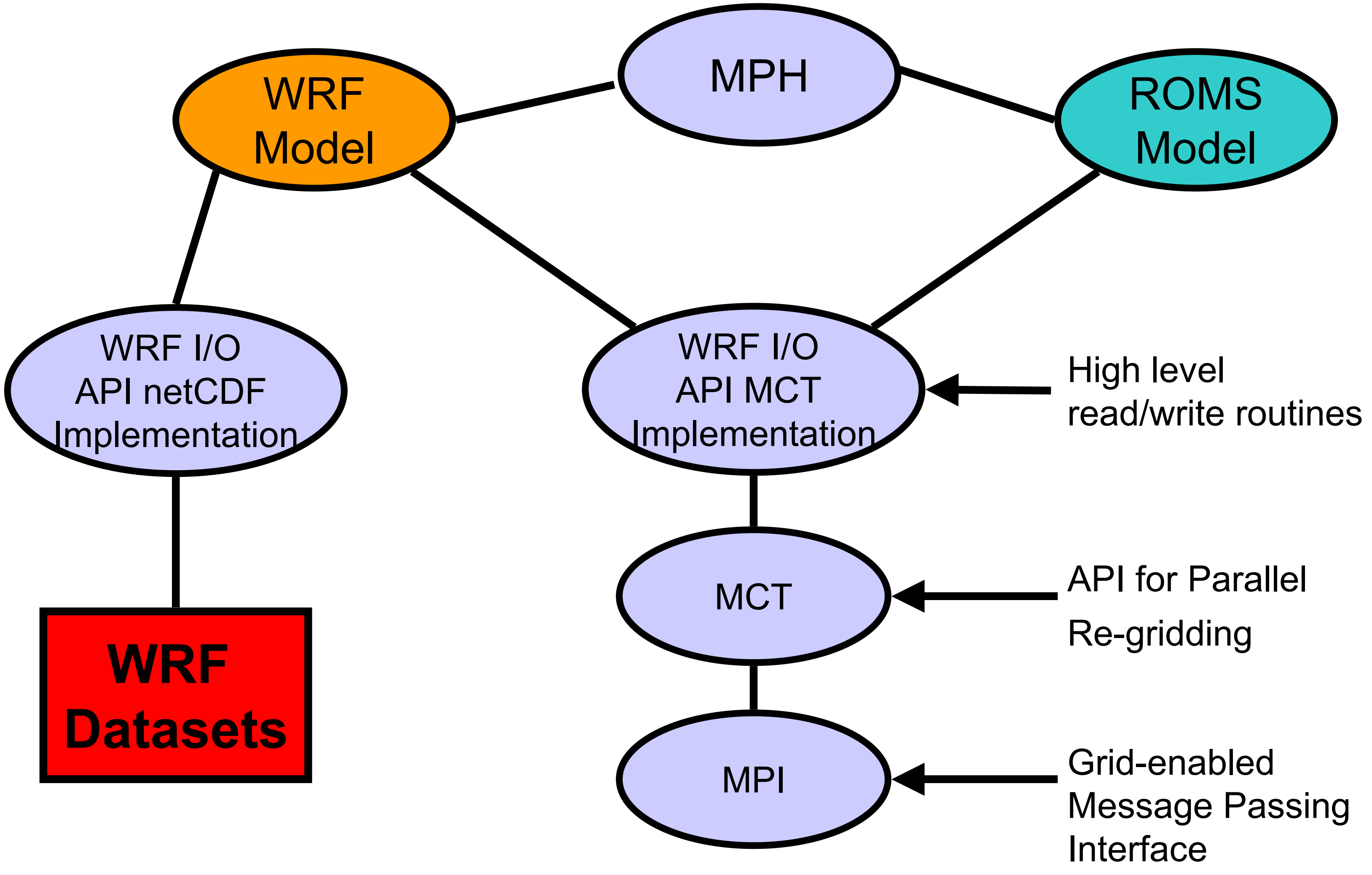
Sample code

```
! Ocean model
call EXT_MCT_IOINIT
call EXT_MCT_OPEN_FOR_READ_BEGIN(“WRF”, Handle)
! Begin “training phase”
call EXT_MCT_READ_FIELD(Handle, U)
call EXT_MCT_READ_FIELD(Handle, V)

! End “Training phase”
call EXT_MCT_OPEN_FOR_READ_COMMIT(Handle)

! Main model loop
! Now actually receive the data
call EXT_MCT_READ_FIELD(Handle, U)
call EXT_MCT_READ_FIELD(Handle, V)
```

Coupled WRF/ROMS implementation and results


MPH
WRF Model
ROMS Model
WRF I/O API netCDF Implementation
WRF I/O API MCT Implementation
High level read/write routines
WRF Datasets
MCT
API for Parallel Re-gridding
MPI
Grid-enabled Message Passing Interface

Coupled WRF/ROMS implementation and results

- WRF
  - Resolution : 150x150x20, time step = 40 seconds
  - Executed for 24 time steps for this test
- ROMS
  - Resolution : 482x482x30, time step = 240 seconds
  - Executed 4 time steps for this test

Coupled WRF/ROMS implementation and results

- ROMS sends SST to WRF
- WRF sends to ROMS

  (5 boundary conditions total):
  - wind stress
  - heat fluxes
  - evaporation – precipitation
- Coupling frequency : every ocean time step

Coupled WRF/ROMS implementation and results

- Use of WRF MCT I/O API produces the correct interpolation
- Performance measured on the FSL Intel Linux cluster
- API performance good, especially since coupling every ocean time step is the worst case scenario

PEs/ model	WRF main loop	WRF receive	WRF send	ROMS main loop	ROMS receive	WRF send
2	107.5	0.73	0.09	158.5	0.63	0.12
4	61.0	0.32	0.02	57.7	0.22	0.07
8	32.9	0.18	0.01	29.5	0.11	0.05
16	19.9	0.13	0.01	13.4	0.05	0.07

Conclusions

- Many design considerations when coupling models
- Free tools available to hide coupling details at various levels
- MCT implementation of WRF I/O API has been used to couple WRF to ROMS
- Implementation performance more than sufficient to enable coupling

Acknowledgements

- John Michalakes – NCAR MMM: WRF I/O API
- Chris Moore – PMEL: Coupled WRF/ROMS Science
- Rob Jacob – Argonne: MCT
- Chris Ding – NERSC: MPH
- Software Infrastructure for Regional Coupled Geophysical Modeling; Michalakes, Schaffer et al. UGC 2003 June 2003
- Chris Ding – LBNL: MPH